

PySAL: The First Ten Years

Sergio J. Rey

January 17, 2019

INTRODUCTION

I am honored to have been selected to give the *Spatial Economic Analysis Plenary*.¹ I would like to use this opportunity to reflect on my experiences as co-founder and lead developer of the Python Spatial Analysis Library: PySAL. I do so as it provides an interesting window on large-scale trends shaping science, of which I think our community of regional scientists need to be more cognizant. My goal in doing so is to uncover what I hope are useful insights as to the importance of the role of community in the functioning of not only academic fields but also in generating new science.

The talk will be a blend of two general themes. The first theme will focus on PySAL and cover its history, the motivation for the project, its structure, and how the library is used. With that point of departure, I will then investigate some larger trends that I think are important. These include the role of the open source movement and the opportunities and challenges it affords regional science, the related development of the open science concept and the challenges it is addressing, the importance of community in fostering innovation in both science and academia, and technological innovations that are growing out of open source and open science which are fundamentally reshaping pedagogy as well as the way we do and report research.

I recognize the structure of this talk is different from the normal presentation of a new set of methods that one might expect in the literature. But I hope that by blending the coverage of the PySAL together with these broader themes, there will be something in it for everybody.

This talk makes three contributions. Through the lens of my experiences on open source projects inside of academia, the underlying institutional barriers that have isolated regional science from the open source revolution are illuminated.

¹This paper is based on the Spatial Economic Analysis Plenary Lecture given at the 58th Congress of the European Regional Science Association, Cork, Ireland, August 29, 2018.

Second, using the specific case of the Python Spatial Analysis Library (PySAL), changes in the way research projects are organized to reap the gains that the open source model offers are examined. Finally, the paper identifies a number of opportunities for regional science to reinvent itself to claim new territory in the modern era of big data.

I first provide an overview of PySAL in terms of the original motivation for the project and its early history. The structure of PySAL and the key spatial analytical functionality it offers are then discussed in Section 3. In section 4, I shift the focus to explore some of the wider currents that my experiences on PySAL and the engagement with open source have identified. These include the rise of open source, open science, open education, and open community movements. Section 5 then brings these trends home to our discipline of regional science where some thoughts on the evolution of our community and the generations of regional scientists identified and the number of opportunities that stand in front of the regional science community.

PYSAL ORIGINS

PySAL began its life as many open source effort do: it was scratching an itch that its creators had. In this case the creators were myself and my long-time colleague Luc Anselin. We have been working in spatial econometrics using the matrix language GAUSS. Around 1993, I was exposed to the Python language through my interactions with the Linux community. After a short span of time with the Python language it became apparent that it offered some major advantages over GAUSS as a language for supporting empirical research. Being a matrix language, GAUSS was excellent for expressing the mathematical structure of different econometrics specification, and for transcribing what one would read in an econometrics textbook into practical code. However, in most empirical research projects the estimation of models represents a tiny fraction of the overall work effort. What often goes unsaid, but is painfully clear, is that much of the effort in a research project is data gathering, munging, harmonization, and transformation. And it was here where GAUSS fell down on its face and Python shined brightly.

Python excelled at string processing and in dealing with heterogeneous data. And although, at the time, the numerical support in Python was much thinner than it is today there were really good third-party modules at that time, for example Numeric was the precursor to today's workhorse numpy. Numeric had a syntax very similar to Matlab's and, in turn, GAUSS which made the transition feasible. The gains Python offered in terms of the data integration and management component of the research process more than offset the relatively weaker

Numerical libraries. And since then the numerical stack in Python has become excellent.

As such I jumped to Python very early. Python was not only a fantastic replacement for GAUSS in terms of doing econometric work, but it had wide scope for all kinds of purposes. I used it to write grading programs, generate indices for textbooks, and even wrote an entire journal management system which I used as my term of editor of the International Regional Science Review. Python quickly became my tool choice for new duties I had to carry out as an academician.

With the move to Python and seeing the changes it led to in our research productivity we started thinking of a collaborative project that built on the work Luc and his students were doing in spatial regression in Python, a package called pySpace, and my work on a package called STARS (Rey and Janikas, 2006). Although the two efforts were targeting different areas of the research stack, spatial econometrics on one hand and space-time data analysis on the other hand, the programs relied on some common constructs in the form of spatial weights and certain types of representations and algorithms. We had been implementing these separately on our own two teams and we thought that, by pooling our development resources and implementing these features in **this is an odd passage "the best way possible score to the best of our abilities" –maybe remove score?**, we could exploit economies of scale. The idea was to build a library of advanced spatial analytical functionality upon which each of our individual projects could then rely to develop specialized applications and, more importantly, a wider Community could utilize.

And that is how PySAL was born.

A second motivation for PySAL stemmed from the recognition that Python was starting to make inroads in other sciences, primarily bioinformatics, astronomy, and physics, yet had seen only limited adoption in regional science and geographic information science. Given the power of the language we had experienced in our own use, we thought this was a missed opportunity and we wanted to do what we could to fuel the uptake of Python in our home disciplines. At the time, there was very limited Python code for doing geospatial analysis, No shapefile readers were available nor anything for spatial data analysis or spatial econometrics.

We found Python to be an excellent language to facilitate rapid prototyping and testing of ideas. It is a simple language to pick up but that does not mean it is a toy language. Not only does it excel from a pedagogical perspective, but can be used to build applications that scale in an impressive fashion. For example Google makes heavy use of Python. There are also prominent scientific projects

that rely on Python. For example, the LIGO project² that recorded the collision of two black holes for the first time or detected gravitational waves made heavy use of Python and its workflow.

The original birth of PySAL, in the sense of our pooling code together to start to build the library, probably dates to sometime around 2007. But as usual, things often take longer than one plans for. The first formal release of PySAL was in July 2010 which came about as both Luc and I had moved to Arizona State University. That move made it clear that geography matters because once we were situated in the same institution it was much easier to organize the project.

Initially we started with a six-month release cycle for PySAL, which aligned very nicely with the academic calendar. We were able to keep to this for the first six years of the project. We are both very proud of that record of releasing every six months for the first six years of the project, on top of all the responsibilities that one has in academia. Looking back, I think this is feasible because PySAL affords many opportunities for structuring independent studies and thesis topics, as well as to organize seminar/studio courses around. I think the same holds for open source projects in general, and I would expect (and hope) that academia becomes home to more such projects.

We were able to leverage these opportunities to benefit our teaching and research goals but also to help the project move forward. I would like to think that we saw this coming in the early days of the library, but actually it is something that emerged with time.

PYSAL STRUCTURE

The original design of PySAL was to have a single monolithic library with sub-components that addressed different types of spatial analysis. This facilitated the easy installation of the package for end-users. Another guiding principle to minimize the complications of the install was fairly restrictive use of dependencies. This ran counter to the normal development philosophy in the open source community where other libraries that had functionality should be relied upon. However, very often in the early days of the library those dependencies were challenging to install, particularly for the target audience of PySAL users who were not developers. What this meant for the developers of PySAL is that we had to roll our own in many cases.

These two features of PySAL served us well in the early days of the project. But as time has passed, the Python spatial analysis stack has matured, we are now at a point where we can start to replace some of the Python implementations

²<https://www.ligo.caltech.edu/>

that the early PySAL team did with more modern and specialized packages for geoprocessing, file reading, and map projections.

Over time we have also come to recognize that the single monolithic architecture of the library, while easing installation, had a number of unintended side effects on the developers. Many of the features in the library were buried deep in lower-level packages. This hindered discoverability of those packages. This meant that the developers of those packages were not getting the recognition that they deserved. This is particularly important in an academic environment where the time dedicated to making these contributions was essentially ignored in tenure and promotion cases. Moreover, the limited discoverability also impacted end users who were not aware of the functionality.

We recently decided to refactor the library to address these two limitations. This has been a major change in the library, taking on the order of two years to implement. The refactoring is recasting PySAL as a meta package which brings together a federation of spatial analytical modules. This has several advantages. Users who may want to focus on, say, spatial econometrics may have no need for all of PySAL, so now they can install `spreg` as its own package. The refactoring also increases discoverability as `spreg` is its own active stand-alone package, and is no longer buried deep inside PySAL. With this increased visibility, adoption increases, leading to greater recognition for the developers as well as more feedback from users and, ultimately, improvements to the package.

From a development perspective the refactoring also increases the speed at which we are able to release new functionality in the individual packages. Previously, under the monolithic model, anytime an enhancement was added to one piece of PySAL a large number of integration tests would be run to ensure that no side effects were triggered by the change. These tests could take on the order of 20 minutes which tended to be frustrating to the developers. Now, with the new packaging model, the developers can run tests that are focused only on their package at hand, and these run much more quickly.³ This increases the cadence of the development for both the individual packages as well as the meta-package.

The other benefit of this model is that end users who still want everything in the PySAL federation can install the meta-package and should notice no difference from their use of the monolithic PySAL package. In other words, we support two different ways for users to interface with the library: users can get everything in one shot through the meta-package, or they could go the a la carte route and pick specific packages in mix and match them to support a specialized workflow.

Since adopting this model, we have also seen benefits in the growth in the number of packages coming into the system. So we are pleased to see that lower-

³There is a meta-package this is responsible for testing the integration of all the PySAL packages.

ing the onboarding cost for new developers has resulted from this refactoring.

Prior to the refactoring there was another major shift in the PySAL Library. We converted from Python2 to Python3 over the course of about a year. Earlier the Python programming language had released a 3.0 version which was not backwards compatible with Python 2. Our approach was to develop in Python 2 to but write converter scripts which would automatically refactor the codebase to Python 3 if a user required Python 3. This was a major effort to implement, and was actually a short-term solution, and a painful one at that. While it supported users who switched to Python 3, it did not allow us to fully exploit the new features in Python 3 as the converted code from Python 2 to had to be backwards compatible. In other words, there are things that one can do in Python 3 that one cannot do in Python 2, so in order to maintain 2.0 backwards compatability we were not be able to take advantage of these Python 3 enhancements. With the refactoring, we have decided to make future versions of PySAL 3.0 only. Users requiring support for Python 2.0 will still be able to use legacy PySAL that will be supported, but only for bug fix releases.

The reorganization of PySAL is along four groups of packages that address the certain type of spatial analysis: explore, model, viz, and lib. Lib is the core package and it is here where we handle file-io, spatial weights, and geoprocessing. All of the other packages in the Python ecosystem import where are dependent upon lib.

Under the explore family of packages we have ESDA which supports exploratory spatial data analysis in the form of global and local test for spatial correlation as well as rates smoothing. GIDDY for geospatial distribution Dynamics implements classic Markov and spatial Markov models for longitudinal spatial data along with measures for spatial income mobility and other types of intra-distributional change. In addition, explore includes spaghetti which is for spatial analysis on networks, and pointpats which supports do you physically **do you physically?** analysis of planar point patterns.

The viz group of packages includes splot, a new package providing a common application programming interface (API) for lightweight visualization functionality on top of the other PySAL packages. mapclassify is a second component of the visualization layer that implements a large number of classification schemes for choropleth mapping, and also supports updating and streaming type data. Rounding out the viz group is legendgram, a novel approach to developing and representing the classification underlying a choropleth map.

The third cluster of packages fall under the model heading. The workhorse here is spreg, which implements modern methods of spatial econometrics and has been a key part of PySAL from day one. As part of the refactoring we have seen much growth in the model space, as new packages that have been added in-

clude mgwr implementing multiscale geographically weighted regression; spint for estimating spatial interaction models, such as the production-constrained or consumption-constrained gravity models; spvcm for spatially-correlated multi-level models; and spglm a package for fitting sparse general linear models (GLM).

Upstream packages that want to use pieces, but not all, of PySAL now have much more flexibility. The most prominent case of this is geopandas⁴ which, prior to the refactoring, would import all of PySAL to have access to the map classification routines. Now as part of the refactoring, the larger import is no longer necessary and geopandas can instead import mapclassify directly so that the dependency footprint is much thinner.

The refactoring has been largely successful, but there are some changes of which longtime users of PySAL should be aware. First, the region module which implemented classical and spatially constrained clustering is no longer part of the meta package. This is due to the development of the standalone package now called region which has a heavy set of dependencies that were produced as part of a Google summer of code project. For the first meta release we have not included region, but users can still install it separately. We have plans to refactor region so that it can be integrated into the PySAL meta-package more easily.

WIDER CURRENTS

PySAL has reached the state that it has because of being embedded in a wider set of developments. There are three currents that have benefited the project. These pertain to the rise of the open-source movement, the development of the open science movement, and the increasing recognition of the importance of scholarly community.

The open source revolution has fundamentally impacted not only science but most aspects of society. Although we may not recognize it directly, the regional science community has benefited from the open source movement. There are two freedoms underlying the notion of free software. First, is the so-called "free as in beer" freedom. This means that there is no monetary cost involved in acquiring software: it is available for anybody who can download it. This has particularly important implications for universities given tight budgets. But this also has profound pedagogical benefits in that students are now no longer tethered to a lab computer holding licensed software. They can now install the software on their own personal computers and time-shift their activity which facilitates greater engagement.

The second, and arguably the more important, freedom is the "free as in free speech" freedom. In general terms, the open source licenses allow users to modify

⁴<https://geopandas.org>

the code directly. From a scientific perspective this is critically important as we will see later, the rise of the open science movement stresses the importance of replication and reproducibility which become all but impossible without access to the scientific source code. The free speech aspect also has important implications for pedagogy in that now users can inspect the source code and demystify the operation of an algorithm. This form of learning provides for a deeper engagement of a student with the underlying computational concepts.

The ability to replicate and reproduce previous research is fundamental to the advancement of science. But building on the shoulders of giants is not possible unless we have access to the shoulders. A slight variation on the theme is that open science, by providing access to the source code and data underlying previous studies, can accelerate scientific discovery. As of now, those source materials can be acquired in a much more expeditious fashion which fuels subsequent studies. This does require a mind-shift on behalf of the scientist who takes the extra steps to release their software and data under open source terms.

It is not only our research production functions that can benefit from adopting open science practices, but our educational efforts can also be enhanced if we borrow from open science and open education developments. In teaching regional science there is so much duplication in individual scholars producing the courses as part of their teaching mission. Everyone goes on it alone and there is limited sharing of materials. At best, perhaps syllabi are exchanged and maybe the occasional PowerPoint is borrowed, but there are no formal mechanisms or any sense of infrastructure to facilitate the sharing. This is changing and other disciplines where entire courses from lecture notes to problem sets are increasingly being posted on open source GitHub repositories. Releasing these materials under Creative Commons license works to protect the intellectual contributions of the original authors and they are very flexible licenses in the sense that they allow for mashing up of the materials with new materials and derivative works.

This type of model is very exciting if one thinks about being able to spend time on an enhancement and building upon the shoulders of a great teacher rather than having to reinvent many teaching wheels. Our courses would be much better if we could start to think about community-based educational materials

The third larger current in which PySAL has swum reflects the growing emphasis placed on the health of a community associated with a project. Here questions about the exclusionary nature of disciplines have been at the forefront many of the open-source meetings that I have attended in past. This has been a highly educational process for me, as I was largely ignorant about the cost to our science of explicit and implicit biases. These biases can lead to different types of barriers to potential community membership. Some of these barriers have been long-standing and are not easily removed, but with sincere and prolonged effort,

I have seen other communities make major strides in redressing these barriers.

BRINGING IT HOME TO REGIONAL SCIENCE

The academic world that I grew up in as a young regional scientist is substantially different than what is emerging now. And part of that emergence is due to the rise of the open-source movement and the changes it has induced in the way science is being organized. There are some opportunities here for regional science.

One key distinction between academia when I was a junior professor and now is that the reward structure is changing. It was very difficult to get recognition for software development contributions. What mattered were journal articles and grants and contracts for promotion cases. As such there was no incentive to pursue those activities and unsurprisingly scientific software for regional science (and all science for that matter, were under-furnished. All the same, I worked on PySAL and related open source projects because I saw the benefits from my own personal research agenda for which these allowed. And I was convinced that these were important activities for me to spend my time on.

Others at the time felt the same way. Jim LeSage was actually doing open source before the term was coined. By releasing his spatial econometrics toolbox⁵ open to researchers, Jim played a major role in stimulating the growth of spatial econometrics.

Paul Waddell's work on the UrbanSim project (Waddell, 2002) is another exemplar of first generation open source regional science. I remember meeting with Paul at 2001 WRSA meeting in Palm Springs and discussing the issues involved in moving UrbanSim from Java to Python. That switch to Python and the explicit open source model for UrbanSim have been a major contributors to the project's success. Clearly, it is an excellent modeling system which is important for its scientific application, but the open source dimension has allowed others to be engaged in its enhancement and evolution, as well as to help drive the adoption of UrbanSim throughout the world.

It is interesting to contrast the environments these first generation of open source regional scientists faced with those that are emerging now. To do so, I highlight the work of three members of this new generation: Dani Arribas-Bel, Levi Wolf, and Geoff Boeing. These individuals are prominent developers on high-profile open source projects and have been very creative in positioning their open source contributions into their formal academic profiles.

Dani has been very generous in posting his Geographic Data Science course⁶

⁵<https://www.spatial-econometrics.com/>

⁶<https://darribas.org/gds16/>

materials on his website and releasing them under Creative Commons licenses⁷. This is incredibly helpful to individuals who are developing similar courses in that those materials are available and do not have to be reinvented. Moreover it is possible to contribute enhancements back to Dani's courses, resulting in a stronger set of materials for future iterations of the course. Dani has been a core developer on the PySAL project where he became introduced open source practices and creatively adopted them to his teaching duties.

Levi Wolf is also a core member of the PySAL development project and has made major contributions to not just to PySAL but other packages in the urban and regional software ecosystem. Chief among these is CenPy⁸, which is an open-source package that allows a researcher to interface with the census api.

Geoff Boeing is a third prominent member of this new generation who has developed the impressive package OSMnx⁹ that facilitates the construction, analysis, and visualization of street networks from OpenStreetMap.¹⁰ It is interesting to note that Geoff worked with Paul Waddell at Berkeley.

In all three cases, we see examples of young regional and urban scientists being exposed to first generation open source projects and then blazing new paths by placing their open source contributions as first class citizens in their evaluation and tenure cases. I think we are fortunate that these individuals are doing this, and that academic institutions are starting to recognize and reward these contributions. As this continues to grow I think we as a community can only benefit as it will bring more members into the discipline as well as improve existing packages and lead to new tools.

These high-profile packages and contributions have brought Dani, Levi, and Geoff increasing recognition as emerging leaders in open source spatial and urban analysis. I am very happy to see these developments as it was never apparent to me that open source would actually succeed in the way it has inside academia. I distinctly remember being told by senior colleagues when I was working on earlier versions of PySAL and the package STARS, that developing tools that are used for research is not research. "You need to be writing papers". My colleagues were being brutally honest with me and were trying to reign in my idealism so that my efforts were more aligned with the realities of promotion and tenure cases at the time. And it is important to note here, that I was really fortunate to be at places where most colleagues were supportive of this work. I often wonder how many of my generation were not so fortunate and did not have the possibility of using some of their research time to do this kind of work.

⁷<https://creativecommons.org/>

⁸<https://github.com/ljwolf/cenpy>

⁹<https://github.com/gboeing/osmnx>

¹⁰<https://www.openstreetmap.org>

Moreover, the climate surrounding open source has changed radically since I was starting in academia. Back then, Microsoft was openly hostile to open source.¹¹ Contrast this with Microsoft's recent development of the Linux subsystem which allows users run native Linux command-line tools directly on Windows.

Indeed, I am optimistic that the tide has turned and we will see more open source regional scientists as we move forward. Being a geographer, however, I cannot fail to notice that there is spatial heterogeneity in this uptake. I was struck by the reception of this talk at the ERSA conference. There was a genuine enthusiasm in the audience for these ideas. While it is the first time I have given this talk, I certainly have mentioned some of these themes elsewhere in papers (Rey, 2017, 2014, 2009) and conferences in the states. It could be that the difference between the excited response at ERSA and the more subdued response in the states may reflect differences in the level of adoption of open source practices in the two regions, with adoption being relatively more advanced in the states and thus the ideas more widely accepted. If true, this would suggest European regional science is ripe for an enhanced engagement with open source practices.

I also want to point out that PySAL itself was first announced to the formal academic world in a regional science journal (Rey and Anselin, 2007). Yet, the uptake of the library has been much more widespread in the GIScience world. I think this reflects the latter being more engaged with developments in machine learning and data science more broadly, while regional science as a field has been fairly slow to explore these areas.

What can we as a academic community do to enhance the adoption of open source practices? We can do a lot. Some of it we are doing already, and I think we should simply continue and enhance these efforts. For example, the NARSC meetings have been offering regular workshops on PySAL and other packages for the past five years. The number of people taking these has continued to expand and, increasingly, the participants are asking asking for multi-day workshops. So the demand is clearly there. This suggests that we should be thinking about more offerings at the regional and international regional science meetings.

The second thing that we can do is to be more welcoming to software development pieces in our regional science journals. As I said, we have already been doing this—in fact, the first paper describing PySAL was published in the *Review of Regional Studies* in 2007. But this has been a rare exception, and since then academicians working on open source software have been looking at different outlets to report these contributions such as the *Journal of Open Source Soft-*

¹¹See the so called Halloween Documents <http://www.catb.org/esr/halloween/>.

ware.¹² While these outlets do provide the authors with academic credit for their contributions, their impact on the field of regional science is limited since these journals are not widely read by academic regional scientists. I would think that our home journals could see this is an opportunity for new types of materials and reinventing their branding in the new era of data science and machine learning.

It is clear that the phrase "data is the new oil" has captured the imagination and spirit of the data science era. And while it is true that data is incredibly valuable to internet companies, I would argue that it is analytics that increases the value of that data. Put another way, if data is the new oil then analytics are the new refineries. And it is here where regional science has huge opportunities. We are all about analytics in the form of models. But I think we need to re-brand ourselves. We bring increasing rigor to the analysis of data in the urban and regional problem domain. Companies are starting to rush in to address this market, however, their underlying analytic frameworks are often proprietary (and therefore of unknown scientific validity), simplistic, or both.

A prominent example of where we are missing opportunities is to compare fantastic visibility of the Gapminder¹³ project by Hans Rosling and colleagues which came up with innovative visualisations of international inequality at the country scale. Contrast this with the massive amount of work that has been done on the question of regional inequality but a complete lack of any high-profile visualization capturing the public's attention to the critical nature of this issue. I think this is low-hanging fruit that could be grasped by a group of regional scientists to help put us back on the radar screen

I am glad to report that I am not Don Quixote here when it comes to the notion of the importance of analytics. My colleague Alan Murray in his 2017 WRSA presidential address (Murray, 2017) actually spoke about the need for regional analytics. We have the raw material, it is a matter of organizing the community around these initiatives. I am fully confident we are capable of doing this and I am very optimistic that we will do so and create an enhanced a more relevant regional science.

REFERENCES

- Murray, A. T. (2017). Regional analytics. *The Annals of Regional Science*, 59(1):1–13.
- Rey, S. J. (2009). Show me the code: spatial analysis and open source. *Journal of Geographical Systems*, 11(2):191–207.

¹²<https://joss.theoj.org/>

¹³<https://www.gapminder.org>

- Rey, S. J. (2014). Open regional science. *Annals of Regional Science*, 52(3):825–837.
- Rey, S. J. (2017). Code as text: Open source lessons for geospatial research and education. In Thill, J.-C. and Dragicevic, S., editors, *Geocomputational analysis and modeling of regional systems*, pages 7–21. Springer.
- Rey, S. J. and Anselin, L. (2007). PySAL: A Python library of spatial analytical methods. *The Review of Regional Studies*, 37(1):5–27.
- Rey, S. J. and Janikas, M. V. (2006). STARS: Space-Time Analysis of Regional Systems. *Geographical Analysis*, 38(1):67–86.
- Waddell, P. (2002). UrbanSim: Modeling urban development for land use, transportation, and environmental planning. *Journal of the American Planning Association*, 68(3):297–314.